



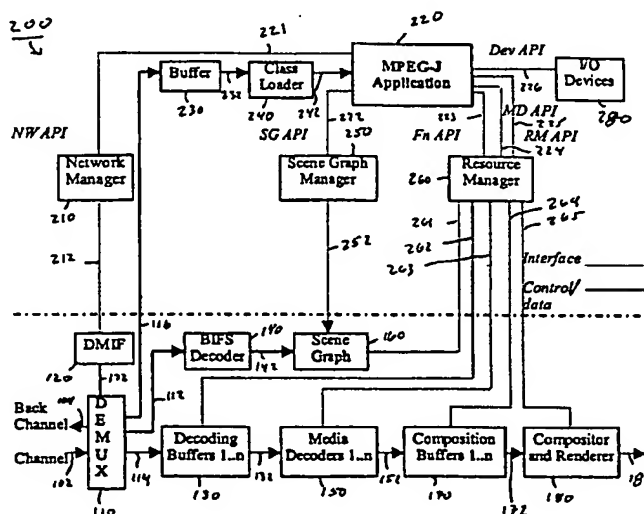
INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 17/30		A2	(11) International Publication Number: WO 00/68840
			(43) International Publication Date: 16 November 2000 (16.11.00)
(21) International Application Number: PCT/US00/12758		(81) Designated States: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).	
(22) International Filing Date: 11 May 2000 (11.05.00)			
(30) Priority Data: 09/309,895 11 May 1999 (11.05.99) US			
(71) Applicants: AT & T CORPORATION [US/US]; 32 Avenue of the Americas, New York, NY 10013-2412 (US). SUN MICROSYSTEMS, INC. [US/US]; 901 San Antonio Road, Palo Alto, CA 94303 (US).			
(72) Inventors: DEUTSCH, Keith, R.; 3192 Maddux Drive, Palo Alto, CA 94303 (US). FERNANDO, Gerard, M.; 142 Waverley Place, Mountain View, CA 94040 (US). SHAH, Pallavi, K.; 448 Kent Drive, Mountain View, CA 94043 (US). SWAMINATHAN, Viswanathan; 5006 Hildasue Terrace, Fremont, CA 94555 (US).			
(74) Agents: DWORETSKY, Samuel, H.; AT & T Corporation, P.O. Box 4100, Middletown, NJ 07748 (US) et al.		Published Without international search report and to be republished upon receipt of that report.	

BEST AVAILABLE COPY

BEST AVAILABLE COPY

(54) Title: ARCHITECTURE AND APPLICATION PROGRAMMING INTERFACES FOR JAVA-ENABLED MPEG-4 (MPEG-J) SYSTEMS



(57) Abstract

MPEG-J is a collection of Java application programming interfaces (APIs) with which applications can be developed to interact with the platform and the content. In the context of MPEG-J, the platform is a device like a set-top box or a PC with Java packages conforming to a well-defined Java platform. The Java-based application consists of Java byte code, which may be available from a local source, like a hard disk, or it may be loaded from a remote site over a network. The MPEG-J Java byte code will be available as a separate elementary stream. The MPEG-4 system is the "Presentation engine" of MPEG-J. MPEG-J provides programmatic control through an "Application engine" which enhances the MPEG-4 browser by providing added interactive capability.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakhstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

ARCHITECTURE AND APPLICATION PROGRAMMING INTERFACES
FOR JAVA-ENABLED MPEG-4 (MPEG-J) SYSTEMS

This non-provisional application claims the benefit of U.S. provisional application No. 60/085,001, entitled "MPEG-J Architecture and API (V0.4)", filed on May 11, 1998, and which is incorporated herein by reference in its entirety. The applicants of the provisional application are Keith Deutsch, Gerard M. Fernando, Atul Puri, Robert L. Schmidt, Pallavi Shah and Viswanathan Swaminathan (Attorney Docket No. Deutsch 1-1-39-23-1-1).

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention is directed to multimedia data storage, transmission and compression systems and methods. In particular, this invention is directed to systems and methods that implement the MPEG-J multimedia data storage, transmission and compression standards. This invention is also directed to control systems and methods that allow for graceful degradation and enhanced functionality and user interactivity of MPEG-4 systems.

2. Related Art

The need for interoperability, guaranteed quality and performance and economies of scale in chip design, as well as the cost involved in content generation for a multiplicity of formats, has lead to advances in standardization in the areas of multimedia coding, packetization and robust delivery. In particular, the International Standards Organization Motion Picture Experts Group (ISO MPEG) has created a number of standards, such as MPEG-1, MPEG-2, MPEG-4 and MPEG-J to standardize bitstream syntax and decoding semantics for coded multimedia.

In MPEG-1 systems and MPEG-2 systems, the audio-video model was very simple, where a given elementary stream covered the entire scene. In particular, MPEG-1 systems and MPEG-2 systems were only concerned with representing temporal attributes. Thus, there was no need to represent spatial attributes in a scene in MPEG-1 systems and MPEG-2 systems.

The success of MPEG-1 and MPEG-2, the bandwidth limitations of the Internet and other distributed networks and of mobile channels, the flexibility of distributed network-based data access using browsers, and the increasing need for interactive personal communication has opened up new paradigms for multimedia usage and control. The MPEG-4 standard addresses coding of audio-visual information in the form of individual objects and a system for combining and synchronizing playback of these objects.

MPEG-4 systems introduced audio-video objects, requiring that the spatial attributes in the scene also need to be correctly represented. Including synthetic audio-video content in MPEG-4 systems is a departure from the model of MPEG-1 systems and MPEG-2 systems, where only natural audio-video content representation was addressed. MPEG-4 systems thus provide the required methods and structures for representing synthetic and natural audio-video information. In particular, MPEG-4 audio-video content has temporal and spatial attributes that need to be correctly represented at the point of content generation, i.e., during encoding, and that also need to be correctly presented at the player/decoder. Because the MPEG-4 player/decoder also allows for limited user interactivity, it should more properly be referred to as an MPEG-4 browser.

Correctly representing temporal attributes in MPEG-4 systems is essentially no different than in MPEG-1 systems and MPEG-2 systems. For these earlier standards, the temporal attributes were used to synchronize the audio portions of the data with the video portions of the data, i.e., audio-video synchronization such as lip-synchronization, and to provide system clock information to the decoder to help buffer management. Because significantly more diverse types of elementary streams can be included in MPEG-4 systems, representing temporal attributes is more complex. But, as mentioned earlier, the fundamental methods for representing temporal attributes in MPEG-4 systems is essentially the same as for MPEG-1 systems and MPEG-2 systems.

In the MPEG-1 systems and MPEG-2 systems standards, the specifications extended monolithically from the packetization layer all the way to the transport layer. For example, the MPEG-2 systems Transport Stream specification defined the packetization of elementary streams (i.e., the PES layer) as well as the Transport layer.

With MPEG-4 systems, this restriction has been relaxed. The transport layer is not defined normatively, as the transport layer is very application specific. It is left to other standards setting bodies to define the transport layer for their respective application areas. One such body is the Internet Engineering Task Force (IETF), which will define standards for transporting MPEG-4 streams over the Internet.

Representing spatial information in MPEG-4 systems is carried out using a parametric approach to scene description. This parametric approach uses the Virtual Reality Modeling Language (VRML). The Virtual Reality Modeling Language allows spatial and temporal relationships between objects to be specified, and allows description of a scene using a scene graph approach. However, in MPEG-4 systems, the Virtual Reality Modeling Language has been extended to provide features otherwise missing from Virtual Reality Modeling Language.

MPEG-4 uses a binary representation, BInary Format for Scene (BIFS), of the constructs central to VRML and extends VRML in many ways to handle real-time audio/video data and facial/body animation. The key extensions to Virtual Reality Markup Language for MPEG-4 systems involve streaming, timing and integrating 2D and 3D objects. These extensions are all included in the BInary Format for Scene (BIFS) specification.

Fig. 1 outlines one exemplary embodiment of a MPEG-4 systems player, which is also referred to as a "Presentation Engine" or an "MPEG-4 browser". The main components on the main data path are the demultiplexer layer, the media decoders, and the compositor/renderer. Between these three sets of components there are decoder buffers and composition buffers, respectively. The MPEG-4 systems decoder model has been developed to provide guidelines for platform developers. The binary format for scene data is extracted from the demultiplexer layer, and it is used to construct the scene graph.

Using application programming interfaces (APIs) has been long recognized in the software industry as a way to achieve standardized operations and functions over a number of different types of computer platforms. Typically, although operations can be standardized via definition of the API, the performance of these operations may still

differ on various platforms, as specific vendors with interest in a specific platform may provide implementations optimized for that platform.

To enhance the features of VRML and to allow programmatic control, DimensionX has released a set of APIs known as Liquid Reality. Recently, Sun
5 Microsystems has announced an early version of Java3D, an API specification that supports representing synthetic audiovisual objects as a scene graph. Sun Microsystems has also released the Java Media Framework Player API, a framework for multimedia playback.

SUMMARY OF THE INVENTION

10 As noted above, when coded multimedia is used for distributed networked and local networked applications on a multimedia data processing system, such as a personal computer, a number of situations may arise. First, the bandwidth for networked access of multimedia may be either limited or time-varying, requiring transmission of only the most significant information, followed by transmitting additional information as more
15 bandwidth becomes available.

Second, regardless of the bandwidth available, the client, i.e., the multimedia data processing system, decoding the transmitted information may be limited in processing and/or memory resources. Furthermore, these resources may be time-varying. Third, a multimedia user may require highly interactive nonlinear browsing and playback. This is
20 not unusual, because significant amounts of textual content on distributed networks, such as the Internet, are capable of being browsed using hyperlinked features and because this is also expected to be true for presentations employing coded audio-visual objects. The parametric MPEG-4 system may only be able to deal with these situations in a very limited way. For example, when the parametric MPEG-4 system is incapable of
25 decoding or presenting all of the coded audio-visual objects, the parametric MPEG-4 system may respond by dropping those objects or temporal occurrences of those objects. However, this results in choppy audio-visual presentations. Further, MPEG-4 may not offer any sophisticated control to the user to allow the user to deal with these situations.

To get around the limitations of this known parametric MPEG-4 system, another
30 known implementation of the MPEG-4 standard is a programmatic MPEG-4 system.

U.S. Patent application serial number 09/055,934, incorporated herein by reference, discloses such a programmatic MPEG-4 system. This programmatic MPEG-4 system includes a set of defined application programming interfaces (APIs) for media decoding, user functionalities and authoring. These application programming interfaces can be invoked by client applications. This programmatic MPEG-4 system allows a number of enhanced real-time and other functions in response to user inputs, as well as graceful degradation in the face of limited system resources available to MPEG-4 clients.

The incorporated 934 application discloses standardized interfaces for MPEG-4 playback and browsing under user control, as well as one type of response to time-varying local and networked resources. These interfaces facilitate adaptation of coded media data to immediately available terminal resources. These interfaces also facilitate interactivity expected to be sought by users, either directly as a functionality or indirectly embedded in audiovisual applications and services expected to be important in the future.

The incorporated 934 application also discloses an interfacing method in the form of a robust application programming interface specification including a visual decoding interface, a progressive interface, a hot object interface, a directional interface, a trick mode, a transparency interface, and a stream editing interface. These interfaces facilitate a substantial degree of adaptivity.

This invention provides systems and methods that use a combination of MPEG-4 media and safe executable code so that content creators can embed complex control mechanisms within their media data to intelligently manage the operation of the audiovisual session.

This invention separately provides systems and methods for implementing the MPEG-J video data storage, compression and decoding standards.

This invention separately provides an improved MPEG-J architecture.

This invention additionally provides an improved MPEG-J architecture having improved structure, modularity and organization.

This invention separately provides an MPEG-J application engine that allows for graceful degradation of MPEG-4 content in view of limited processing, memory or bandwidth resources.

This invention separately provides an MPEG-J application engine that allows for enhanced functionality of and user interactivity with MPEG-4 content.

This invention separately provides application programming interfaces for MPEG-J.

5 New paradigms in communication, software and networking, such as that offered by the Java™ language, offer new opportunities for flexibility, adaptivity and user interaction. For instance, the advent of the Java™ language offers networking and platform independence critical to downloading and executing of applets, such as, for example, Java classes, on a client system from a server system storing the applets.

10 Depending on the design of the applet, either a single access to the data stored on the server may be needed and all the necessary data may be stored on the client, or several partial accesses may be needed. This partial access design is used to reduce storage space and time needed for startup. This partial access design is referred to as streamed playback.

15 This invention provides a collection of Java API's with which applications can be developed to interact with a data processor and content. According to this invention, MPEG-J is a Java™-enabled set of standards that define the file organization, storage and compression of video data streams. In the context of MPEG-J according to this invention, the data processor can be implemented as a set-top box or a PC with Java

20 packages conforming to a well-defined Java platform. The Java-based application includes Java byte code, which may be available from a local source, like a hard disk, or which may be loaded from a remote site over a network. As indicated above, the term "MPEG-4 browser" refers to the MPEG-4 system. MPEG-J adds programmatic control to the MPEG-4 system, through an "Application engine". The Application Engine

25 enhances the Presentation Engine by providing added interactive capability. The MPEG-J Java byte code will be available to the MPEG-J Application engine as a separate elementary stream.

The improved architecture and application programming interfaces (APIs) of MPEG-J according to this invention allow selective media decoding facilitating graceful

degradation to varying resources of a client, as well as improved functionalities as required in interactive user applications.

In one potential use of MPEG-J, a content provider designs all of the MPEG-J content, i.e., the MPEG-J data stream. This use of the MPEG-J standard is desirable for content providers, and requires only incremental updates to the MPEG-J data stream. In this case, any changes to the MPEG-J data stream can be done using binary format for scene (BIFS) updates. In another potential use of MPEG-J, the client dynamically controls the displayed video scene generated from the MPEG-J data stream. This use of the MPEG-J standard is desirable for set top manufacturers. However, change in non-updatable nodes of the MPEG-J data stream may not be possible.

MPEG-J will eventually need to serve both of the potential uses. Since the first potential use is more deterministic than the second potential use, the systems and methods of this invention fully implement the first potential use. The systems and methods of this invention provide the hooks to partially implement the second potential use. In particular, in one exemplary embodiment of the systems and methods of this invention, the MPEG-J scene graph capabilities are always based on the tightly-integrated, i.e., content-provider-oriented, model.

In one exemplary embodiment of the application programming interfaces of this invention, the MPEG-J application programming interface (API) is not a single application programming interface, but rather is a collection of application programming interfaces (APIs) that address various interfaces for a flexible MPEG-4 system. In one exemplary embodiment, the MPEG-J application programming interfaces are implemented using the Java language. The application programming interfaces include one or more of an Application Manager API; a SceneGraph Manager API; a Resource Manager API; an Interactivity Manager API; a Media Decoders API; a Functionality API; a Networking API; and a Device API. In one exemplary embodiment of the application programming interfaces of this invention, one or more of these application programming interfaces are implemented as object-oriented-programming object classes. The object classes are organized into various packages. In particular, various objects of one or more of the application programming interfaces are collected into a single package.

These and other features and advantages of this invention are described in or are apparent from the following detailed description of various exemplary embodiments of the systems and methods according to this invention.

BRIEF DESCRIPTION OF THE DRAWINGS

5 Various exemplary embodiments of this invention will be described in detail, with reference to the following figures, wherein:

Fig. 1 is a functional block diagram of one exemplary embodiment of an MPEG-4 browser;

10 Fig. 2 is a functional block diagram of one exemplary embodiment of the components of an MPEG-J operating environment according to this invention;

Fig. 3 is a functional block diagram showing in greater detail a first exemplary embodiment of the MPEG-J application engine of Fig. 2 according to this invention;

Fig. 4 is a functional block diagram showing in greater detail a second exemplary embodiment of the MPEG-J application engine of Fig. 2 according to this invention; and

15 Fig. 5 is a functional block diagram showing in greater detail a third exemplary embodiment of the MPEG-J application engine of Fig. 2 according to this invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Fig. 1 is a functional block diagram outlining one exemplary embodiment of a MPEG-4 browser or presentation engine 100. As shown in Fig. 1, the presentation
20 engine 100 includes a demultiplexer layer 110, a Delivery Multimedia Integration Framework (DMIF) subsystem 120, one or more decoder buffers 130, a Binary Format for Scene (BIFS) decoder 140, one or more media decoders 150, a scene graph controller 160, one or more compositors 170, and a compositor/renderer 180. In particular, audio-video data is input to the demultiplexer layer 120 over a channel signal line 102, while
25 handshake data is output by the demultiplexer layer 120 over a backchannel signal line 104.

The demultiplexer layer 110 is connected to the delivery multimedia integration framework subsystem 120 over a data channel 122. The demultiplexer layer 110 is also connected to the binary format for scene decoder 140 over a data channel 112 and to each
30 decoder buffer 130 over a data channel 114. The binary format for scene decoder 140 is

connected to the scene graph controller 160 over a data channel 142. Each decoder buffer 130 is connected to a corresponding media decoder 150 over a data channel 132. Each media decoder 150 is connected to a corresponding compositor buffer 170 over a dedicated data channel 152. Each compositor buffer 170 is connected to the
5 compositor/renderer 180 over a data signal channel 172. The compositor/renderer 180 is connected over a data channel 182 to a downstream processing subsystem, such as a display.

The scene graph controller 160 outputs control signals to the compositor/renderer 180 over a control channel 162. The scene graph controller also outputs control signals to
10 each media decoder over the channel or channels 162.

In operation, scene content in the form of MPEG-4 encoded audio-video streams is input to the demultiplexer layer 110 over the channel signal line 102. The encoded audio-video streams include a scene graph stream. The demultiplexer layer 110 uses the backchannel signal line 104 to acknowledge receipt of data, or to request data be resent,
15 and, in a client-server environment, sends commands back to the server, and the like. The demultiplexer layer 110 then demultiplexes the encoded audio-video streams based on data input from the delivery multimedia integration framework subsystem 120 over the data channel 122, and outputs the demultiplexed scene graph stream, comprising encoded binary format for scene data, to the binary format for scene decoder 140 over the
20 data channel 112 and each of the other encoded audio-video streams an appropriate one of the decoder buffers 130 over a corresponding one of the data channels 114. The binary format for scene decoder 140 decodes the encoded binary format for scene data and outputs it to the scene graph controller 160 over the data channel 142.

The scene graph controller 160 inputs the decoded binary format for scene data
25 and generates a scene graph from the input binary format for scene data. The scene graph defines which objects will appear in the rendered scene, based on the binary format for scene data associated with each node of the generated scene graph. The scene graph controller 160, based on the generated scene graph, outputs decoding control signals over the channel or channels 162 to one or more of the media decoders 150.

Based on the decoding control signals from the scene graph controller 160, one or more of the media decoders 150 inputs encoded audio-video data appropriate for that media decoder 150 from the corresponding decoder buffer 130 over the corresponding data channel 132. Each such media decoder 150 decodes the input audio-video data to
5 generate one or more audio or video objects. The decoded audio objects or video objects are output by that media decoder 150 to the corresponding compositor buffer 170 over the corresponding data channel 152.

The compositor buffers 170 store the decoded audio and video objects generated by the media decoders 150. The compositor/renderer 180, based on rendering control
10 signals output by the scene graph controller 160 over the control channel 164, inputs various audio and video objects stored in the compositor buffers 170 over the data channels 172 and composes the input audio and video objects into a scene. In particular, the scene graph controller 160 generates the rendering control signals based on a current location within the scene graph. The compositor/renderer 180 outputs the rendered scene
15 to a downstream processing subsystem, such as a display subsystem that displays the rendered scene on a display device. The compositor/renderer 180 may respond to very basic user interactions, such as mouse clicks, etc.

In one exemplary embodiment of the MPEG-4 browser 100 shown in Fig. 1, MPEG-4 coded data read from a storage device or received over a distributed network,
20 such as a LAN, a WAN, the Internet or an intranet, goes through the delivery multimedia integration framework subsystem 120 and the demultiplexer layer 100. In the demultiplexer layer 100, FlexMux-PDU's pass through a Sync Layer resulting in unformatted SL-PDUs of each media type, such as, for example, encoded audio streams, encoded video streams, encoded speech streams or encoded facial animation streams.
25 The unformatted SL-PDUs of each media type are then buffered in the respective decoder buffers 130 and are offered to the corresponding media decoders 150. Also, any SL-PDUs corresponding to scene description representation are input to the binary format for scene decoder 140. The output of the binary format for scene decoder 140 goes to the scene graph controller 160. The output of the media decoders 150 as well as the scene

graph controller 160 feeds the compositor/renderer 180. The output of the compositor/renderer 180 is the scene for presentation.

The MPEG-J systems and methods according to this invention are flexible programmatic control systems and methods that represent an audio-visual session in a manner that allows the session to adapt to the operating characteristics of the multimedia data processing system on which the MPEG-J application, the MPEG-4 browser and an execution engine, such as a display subsystem, are executing. The MPEG-J systems and methods according to this invention allow graceful degradation under limited or time varying resources. The MPEG-J systems and methods according to this invention also provide the ability to respond to user interaction and provide enhanced multimedia functionalities.

The MPEG-J systems according to this invention are programmatic systems that specify interfaces for inter-operation of an MPEG-4 media browser with Java code. By combining MPEG-4 media and safe executable code, content creators may embed complex control mechanisms with their media data to intelligently manage the operation of the audio-visual session. In the MPEG-J systems according to this invention, Java code is delivered as a separate elementary stream to the multimedia data processing system. The Java code is directed to a MPEG-J run time environment of the multimedia data processing system. The MPEG-J run time environment includes a Java Virtual Machine, which provides the Java code access to the various components of the MPEG-4 browser 100.

Fig. 2 is a functional block diagram of one exemplary embodiment of the components of the MPEG-J operating environment. As shown in Fig. 2, a multimedia data processing system 10 includes an MPEG-J application engine 200, the MPEG-4 browser or presentation engine 100 and the execution engine 190. The MPEG-J application engine 200 includes an MPEG-J application 220 and an MPEG-J run time environment 20. The MPEG-J run time environment 20 includes a Java Virtual Machine 22 and a number of MPEG-J application programming interfaces (MPEG-J APIs) 24, as well as the standard Java system files 26.

The MPEG-J systems and methods of this invention deliver and run application programs that affect the behavior of the MPEG-4 browser or presentation engine, the rendered scene, and the content programming in general. To that end, the MPEG-J systems and methods of this invention define application program interfaces (APIs) to the various components of the MPEG-4 browser or presentation engine 100. Further, the MPEG-J systems and methods of this invention describes the delivery of application programs using the standard MPEG-4 elementary streams delivery mechanism. Besides delivering application programs to the MPEG-4 browser or presentation engine, the MPEG-4 browser or presentation engine can be configured with local application programs that use the same API's. Examples of sources of such local programs are programs that are downloaded, as well as commercially available programs that enhance the performance and functionality of the MPEG-4 browser or presentation engine. It should be appreciated that local programs will generally have no knowledge of the delivered specific MPEG-4 content. In contrast, the delivered programs that originate from the same author as the audio/visual MPEG-4 content will generally have knowledge of the delivered specific MPEG-4 content.

The architecture of the MPEG-J application engine 200 takes into consideration the resources available on the underlying multimedia data processing system 10. The architecture of the MPEG-J run time environment 20 isolates distinct components of the MPEG-J application engine 200, the design of application programming interface that reflects these components, and the characterization of interactions between these components.

The distinct components of the MPEG-J run time environment 20 include execution and presentation resources, decoders, network resources, and the scene graph. The architecture of the MPEG-J application engine 200 assumes that decoding and presenting resources of the underlying multimedia data processing system 10 are limited. The execution and presentation resources components abstract access to information on such static and dynamic resources in the MPEG-4 browser or presentation engine and notification during changes in such resources. The execution and presentation resources

components provides for some minimal control of the static and dynamic resources in the MPEG-4 browser or presentation engine.

5 The decoders component abstracts the media decoders used to decode the received media streams. The decoders component also abstracts the programmatic control of the media decoders and the manipulation of the media decoders that is used to add extra functionality to the MPEG-4 browser or presentation engine.

10 As described above, the MPEG-4 browser or presentation engine receives media streams. The network resources component abstracts the control of the received media streams. The network resources component also abstracts the media pipeline or pipelines that transport and present the media streams to the MPEG-4 browser or presentation engine.

15 As described above, each session presented by the MPEG-4 browser or presentation engine has a Scene Graph to represent and spatially and temporally locate the audio-visual objects of that session. The scene graph component abstracts access to and control of the scene graph.

20 Fig. 3 shows a functional block diagram illustrating in greater detail a first exemplary embodiment of the MPEG-J application engine 200 of Fig. 2 according to this invention. The MPEG-J systems and methods according to this invention allows a combination of MPEG-4 media and safe executable code, so that the content creators can embed complex control mechanisms within their media data to intelligently manage the operation of the audio-visual session. In the first exemplary embodiment shown in Fig. 3, the architecture of the MPEG-J systems according to this invention includes, in the lower portion of the MPEG-J run time environment 20, the exemplary embodiment of the MPEG-4 browser or presentation engine 100 shown in Fig. 2. The MPEG-J system or application engine 200 forming the upper portion of the MPEG-J run time environment 25 specifies the interfaces and control structures that control the MPEG-4 browser or presentation engine 100.

30 As shown in Fig. 3, the first exemplary embodiment of the MPEG-J system or application engine 200 includes a network manager 210, the MPEG-J application controller 220, a buffer 230, a class loader 240, a scene graph manager 250, a resource

manager 260, and one or more input/output (I/O) devices 280. The I/O devices 280 can be any known or later developed input or output device, including a mouse, a keyboard, a touchscreen, a pen device, a set-top remote control device, a television remote control device, a display device, such as a television, a CRT-type display, an LCD-type display, an LED-type display, a printer, or the like.

The network manager 210 is connected to the delivery multimedia integration framework subsystem 120 and the MPEG-J application controller 220 through interface channels 212 and 221, respectively, of a network application programming interface Nw API. The buffer 230 is connected by a data channel 116 to the demultiplexer layer 110 and a data channel 232 to the class loader 240. The class loader 240 is connected to the MPEG-J application controller 220 by a data channel 242. The scene graph manager 250 is connected to the scene graph controller 160 and the MPEG-J application controller 220 through interface channels 252 and 222, respectively, of a scene graph application programming interface SG API.

The resource manager 260 is connected to the MPEG-J application controller 220 through interface channels 223, 224 and 225 of a functionality application programming interface Fn API, a media decoders application programming interface MD API, and a resource manager application programming interface RM API, respectively. The resource manager 260 is also connected through interface channels 261-265 of the resource manager application programming interface RM API to the scene graph manager 160, the decoder buffers 130, the media decoders 150, the compositor buffers 170 and the compositor/renderer 180, respectively. The I/O devices 280 are connected to the MPEG-J application controller 220 through an interface channel 226 of a device application programming interface Dv API. The network application programming interface Nw API, the scene graph application programming interface SG API, the functionality application programming interface Fn API, the media decoders application programming interface MD API, the resource manager application programming interface RM API, and the device application programming interface Dv API are described in greater detail below.

The network manager 210 interfaces the MPEG-J application controller 220 with the delivery multimedia integration framework subsystem 120 using the interface channels 212 and 221 of the network application programming interface Nw API. The scene graph manager 250 interfaces the MPEG-J application controller 220 with the scene graph controller 160 using the interface channels 222 and 252 of the scene graph application programming interface GA API. The resource manager interfaces the MPEG-J application controller 220 with the scene graph controller 160, the decoder buffers 130, the media decoders 150, the compositor buffers 170 and the compositor/renderer 180 using the interface channels 261-265 of the resource manager application programming interface RM API. The interface channel of the device application programming interface Dv API allows users to directly interact with and control the MPEG-J application controller 220 using one or more of the I/O devices 280. Thus, the user input, through the MPEG-J application controller 220, can be used to control the various elements of the MPEG-4 browser or presentation engine 100.

The scene graph manager 250 allows direct control of the binary format for scene scene graph that specifies the scene to be rendered. In particular, the scene graph manager 250 is able to turn off portions of the scene graph to better match the processing and/or memory resources of the underlying multimedia data processing system 10 available to the MPEG-J run time environment 20 to the resources needed to render the scene. Thus, the scene graph manager controls the scene graph controller to modify the scene graph or the generated multimedia scene in view of the current system resources, user inputs, and/or any embedded control mechanisms, such as behavior control commands. In particular, the scene graph or the generated multimedia scene is modified to gracefully or controllably degrade the generated multimedia scene.

The resource manager 260 manages system resources of the underlying multimedia data processing system 10, allowing the regulation of overall performance. This includes controlling the scene graph controller 140, the media buffers 130, the media decoders 150, the compositor buffers 160, the compositor/renderer 170 and MPEG-J run time environment 20 functionalities and resources in view of the current system resources, user inputs, and/or any embedded control mechanisms, such as behavior

control commands. In particular, the resource manager 260 controls these elements to gracefully or controllably degrade the generated multimedia scene.

The network manager 210 provides a means for the MPEG-J application controller 220 to send requests or receive data through the MPEG-J application controller 220. The network manager 210 can also monitor the network resources of the distributed network, and can control the connection to the distributed network in view of the current network resources, user inputs, and/or any embedded control mechanisms, such as behavior control commands. The network manager 210 also allows the MPEG-J application controller 220 access to the network resources of the distributed network.

The MPEG-J application controller 220 can then control the MPEG-4 browser or presentation engine 100 in view of the current network resources, user inputs, and/or any embedded control mechanisms, such as behavior control commands.

Fig. 4 shows a functional block diagram illustrating in greater detail a second exemplary embodiment of the MPEG-J application engine 300 of Fig. 2 according to this invention. The second exemplary embodiment of the MPEG-J application engine 300 is generally identical to the first exemplary embodiment of the MPEG-J application engine 200 shown in Fig. 3, except that the second exemplary embodiment of the MPEG-J application engine 300 separates the control and data channels and reorganizes the interface and control channels between a number of the components of the second exemplary embodiment of the MPEG-J application engine 300. In the second exemplary embodiment of the MPEG-J application engine 300, the network manager 210 has been deleted, with the network interface channel 321 connected directly to the delivery multimedia integration framework subsystem 120. The MPEG-J application controller 320 includes an application manager 390.

As shown in Fig. 4, the scene graph manager 350 is still connected to the scene graph controller 160 and the MPEG-J application controller 320. However, the scene graph manager 350 is now connected to the scene graph controller 160 over a control channel 351 and to the MPEG-J application controller 320 over an interface channel 322 of a scene graph-external application interface application programming interface SG-EAI API. The scene graph controller 160 is now connected to the application manager

390 over an interface channel 391 of an application manager application programming interface AM API.

Similarly, the resource manager 360 remains connected to the MPEG-J application controller 320 through interface channels 323 and 325 of the functionality application programming interface Fn API and the resource manager application programming interface RM API, respectively. However, the resource manager 360 is now connected to the application manager 390 over the interface channel 324 of the media decoders application programming interface MD API. Furthermore, the resource manager 260 is now connected through control channels 361-365 to the scene graph manager 160, the decoder buffers 130, the media decoders 150, the compositor buffers 170 and the compositor/renderer 180, respectively.

The network application programming interface Nw provides a means for the MPEG-J application controller 220 to send requests or receive data through the MPEG-J application controller 220. The MPEG-J application controller 220 can also monitor the network resources of the distributed network, and can control the connection to the distributed network in view of the current network resources, user inputs, and/or any embedded control mechanisms, such as behavior control commands. The network application programming interface Nw also allows the MPEG-J application controller 220 access to the network resources of the distributed network. The MPEG-J application controller 220 can then control the MPEG-4 browser or presentation engine 100 in view of the current network resources, user inputs, and/or any embedded control mechanisms, such as behavior control commands.

Fig. 5 shows a functional block diagram illustrating in greater detail a third exemplary embodiment of the MPEG-J application engine 400 of Fig. 2 according to this invention. The third exemplary embodiment of the MPEG-J application engine 400 is generally identical to the second exemplary embodiment of the MPEG-J application engine 300 shown in Fig. 4, except that an interactivity manager 470 is positioned between the MPEG-J application controller 420 and the I/O devices 480, and the scene graph manager 450 is not directly connected to the scene graph controller 160. As shown in Fig. 5, the MPEG-J application controller 420 is connected to the interactivity manager

470 over an interface channel of an interactivity manager application programming interface IM API, while the interactivity manager 470 is connected to the I/O devices 480 over the device interface channel of the device application programming interface Dv API.

5 Likewise, the scene graph manager 450 is no longer connected to the scene graph controller 160 over the interface channel 452 of the scene graph application programming interface SG-EAI API. As in the second exemplary embodiment of the MPEG-J application engine 300 shown in Fig. 4, the scene graph controller 160 is connected to the application manager 490 over the interface channel 391 of the application manager
10 application programming interface AM API. However, the scene graph manager 450 is now connected to a switch 456 over a data/control channel 453 of the scene graph application programming interface SG-EAI API.

 The A channel out of the switch 456 directs the control and/or data signals from the scene graph manager 450 to a binary format for scene encoder 454. The binary
15 format for scene encoder 454 inputs the data and control signals from the scene graph manager 450 and converts them into new scene content. The binary format for scene encoder 454 outputs the new scene content over a data channel 457 to the binary format for scene decoder 140, where the new scene content is decoded indistinguishably from original scene content and output over the data channel 142 to the scene graph controller
20 160.

 The B channel out of the switch 456 directs the control and/or data signals from the scene graph manager 450 to the scene graph controller 160 over a data channel 458. Thus, while the scene graph manager 450 continues to control the scene graph controller 160, the scene graph manager 450 does not directly interface with the scene graph
25 controller 160.

 However, it should be understood that the binary format for scene encoder 454 does not have to be implemented in the third exemplary embodiment of the MPEG-J application engine 400. Rather, as in the first and second exemplary embodiments of the MPEG-J application engine 200 and 300, the scene graph manager 450 can be directly

connected to the scene graph controller 160 over the interface channel 452 of the scene graph application programming interface SG-EAI API.

5 The interactivity manager application programming interface IM API and the interactivity manager 470 permit a greater degree of user input and interactivity with the multimedia scene generated by the MPEG-4 browser or presentation engine 100. Thus, the user input, through the MPEG-J application controller 220, can be used to control and interact with the various elements of the MPEG-4 browser or presentation engine 100 to a greater extent.

10 The MPEG-J application engines 200-400 outlined above are specific exemplary embodiments of a multimedia browsing control subsystem of a multimedia browsing system according to this invention. The MPEG-4 browser or presentation engine 100 outlined above is a specific exemplary embodiment of a multimedia browsing subsystem of a multimedia browsing system according to this invention. The multimedia browsing subsystem generates a multimedia scene based on a plurality of data streams. The data streams include data defining a number of multimedia objects and data defining control programs or mechanisms that are associated with the multimedia objects.

15 The multimedia browsing control subsystem (e.g., the MPEG-J application engines 200-400) according to this invention controllably degrades (i.e., graceful degradation of) the multimedia scene generated by the multimedia browsing subsystem (e.g., the MPEG-4 browser or presentation engine 100) in the face of limited system resources available to the multimedia browsing subsystem, and/or in view of complex control mechanisms, or behavior control commands, embedded within the data defining the multimedia scene for intelligently managing the generation of the multimedia scene. The multimedia browsing control subsystem according to this invention separately enables a number of enhanced real-time and other functions for controllably degrading and interacting with the multimedia browsing subsystem in response to user inputs.

20 It should be understood that each of the buffers, managers, controllers, decoders, application programming interfaces and other elements of the multimedia browsing systems shown in each of Figs. 1-5 can be implemented as portions of a suitably programmed general purpose computer used to implement the multimedia data

30

processing system 10. Alternatively, each of the elements shown in each of Figs. 1-5 can be implemented as physically distinct hardware circuits within an ASIC, or using a FPGA, a PDL, a PLA or a PAL, or using discrete logic elements or discrete circuit elements, depending on the particular hardware used to implement the multimedia data processing system 10. The particular form each of the circuits shown in each of Figs. 1-5 will take is a design choice and will be obvious and predicable to those skilled in the art. For example, the MPEG-J application engine and the MPEG-4 presentation engine can each be implemented as software executing on a programmed general purpose computer, a special purpose computer, a microprocessor or the like that is used to implement the multimedia data processing system 10. In this case, the MPEG-J application engine and the MPEG-4 browser can each be implemented as an embedded routine, as a resource residing on a server, or the like.

The channel 102 and the back channel 104 can be implemented using any known structure or apparatus for transmitting the MPEG-encoded data to the MPEG-4 browser. Thus, the channel 102 and the back channel 104 can be implemented using a public switched telephone network, a local or wide area network, an intranet, the Internet, a wireless transmission channel, any other distributing network, or the like.

There are two scenarios in which MPEG-J may be used. In one possible scenario, the content provider designs all MPEG-J content, and all of the necessary or desirable add-ons for local use and interactivity, such as including commercials, local programming information, scene interactivity, and other subsequent modifications. This is desirable for content providers, as it allows incremental updates. That is, change of state can be done via binary format for scene updates. In a second possible scenario, the client dynamically controls the scene. This is desirable for set-top manufacturers. It should also be appreciated that local programming and interactivity can also be provided by an intermediary between the original content provider and the client.

However, it should be appreciated that changes in non updatable nodes may not be possible. The MPEG-J systems and methods of this invention may eventually need to serve both of these scenarios. Since the first scenario is more deterministic than the second scenario, the exemplary embodiments of the MPEG-J systems and methods of this

invention that are described herein mainly support the first scenario while including the hooks to partially support the second scenario.

Application programs based on the MPEG-J API's are made available to the MPEG-4 browser or presentation engine 100 in the form of an MPEG-4 elementary stream. This implies that the "MPEG-J elementary stream" has an associated Elementary Stream Descriptor. Further, a stream type "MPEG-J Stream" is defined in the StreamType table (Table 9) in ISO/IEC 14496-1. The MPEG-J data could be classes or serialized objects transmitted as byte code from servers to clients. Serialized objects are expected to accompany classes that have knowledge about handling those objects.

An MPEG-J session is initiated when the MPEG-4 browser or presentation engine 100 receives an MPEG-J object descriptor. In response to receiving the MPEG-J object descriptor, the MPEG-4 browser or presentation engine 100 opens an MPEG-J elementary stream via a DMIF channel in the demultiplexer layer 110. The MPEG-J elementary stream is an SL packetized stream, similar to all other MPEG-4 streams. The MPEG-4 browser or presentation engine 100 then delivers the access units of the MPEG-J elementary stream to the buffer 230 of the class loader 240 over the data channel 116. The class loader 240 loads the classes.

An MPEG-J application that uses the MPEG-J application programming interfaces of this invention to control the MPEG-4 browser or presentation engine 100 can either be local or remote relative to the underlying multimedia data processing system 10. In the case of a remote application that is received in the MPEG-J elementary stream, the remote application must implement a remote application interface. The header to the remote application and the remote application interface can be implemented using any appropriate and/or relevant techniques and programming structures known to those skilled in the art.

MPEG-J scene graph capabilities are always based on the tightly integrated or content provider oriented model. In general, the first stream received is the "master" scene graph. Subsequent streams are added as "subtrees" at designated nodes in the master scene graph. The "master" scene graph must acquire its subtrees. However, any tree can be the master scene graph. Nonetheless, subtrees inherit acquisition limits.

The MPEG-J application programming interface is not a single application programming interface but a collection of application programming interfaces that address various interfaces for a flexible MPEG-4 system. Java language is used for application programming interface specification. The MPEG-J application engine
5 includes a number of categories of application programming interfaces, as outlined above. These categories of application programming interfaces include the application manager application programming interface; the scene graph manager application programming interface; the resource manager application programming interface; the interactivity manager application programming interface; the media decoder application
10 programming interface; the functionality application programming interface; the networking application programming interface; and the device application programming interface. Table 1 outlines these application programming interfaces and their main classes or interfaces:

Table 1

API Category	Main Classes Or Interfaces	Explanation
Application Manager	Application Manager	Direct interface to scene graph to deal with properties
Scene Graph Manager	SBrowser	Means by which MPEG-J apps access and manipulate the BIFS player
Resource Manager	Resource Manager	Centralized facility for managing system resources
Interactivity Manager	InteractivityManager	Centralized facility for managing user interactivity with scene
Media Decoder	MPDecoder	Access and control to the decoders used to decode the audio-visual objects.
	Video Decoder	Decoding of video objects
	TextureDecoder	Decoding of image texture objects
	MeshDecoder	Decoding of wireframe mesh objects
	FaceDecoder	Decoding of Face objects
	AudioDecoder	Decoding of speech and Audio objects
	TTSDecoder	Decoding of TTS objects
	StructAudioDecoder	Decoding of structured audio objects
Functionality	ProgressiveImage	Progressive decoding of image texture object under user control
	HotObject	Selective decoding of active objects for enhancement under user control
	DirecDecoding	Selective decoding of object based on viewpoint under user control
	Transparency	Selective decoding of full or partial objects based on transparency and user control
	TrickMode	Selective decoding of object for playback in trick mode (FF,FR) under user control
	Adjust Audio	Selective decoding and enhancement of audio object under user control
Network	NetworkManager	Access and control of the network components of the MPEG-4 browser.
	BackChannel	Access and control of back channel
Device	Input Device	An interface for input device drivers
	UserInputDevice	An interface for user input device drivers
	AudioDevice	An interface for audio devices
	SmartCard	An interface for smart card devices

List of MPEG-J Application Programming Interfaces

Packages are a means to organize the implementations of application programming interfaces. For the list of MPEG-J application programming interfaces outlined above for the systems and methods of this invention, one exemplary set of packages includes an "mpgj.sys" package, an "mpgj.dec" package, an "mpgj.func" package, and an "mpgj.dev" package. The "mpgj.sys" package contains classes for application, scene graph, resource and interactivity manager. The "mpgj.dec" package contains classes for Media decoding. The "mpgj.func" package contains classes for functionalities including interaction. The "mpgj.dev" package contains classes that provide interfaces to various input, output, sound and video devices.

A local or remote MPEG application will use well-defined sets of MPEG-J application programming interfaces to interact with the underlying MPEG-4 browser or presentation engine 100. These Java™ application programming interfaces are implemented on all compliant MPEG-4 browsers or presentation engines. The details of the required implementation of these application programming interfaces can be found in the MPEG-4 systems Version 1 Standard (ISO/IEC 14496-1). It is expected that the local or remote MPEG-J application engine can call methods defined in the MPEG-J application programming interfaces and those from a minimal set of Java™ packages. The MPEG-J application programming interfaces are functionally divided into a network (Nw) application programming interface category; a resource manager (RM) application programming interface category; a media decoder (MD) application programming interface category; a scene graph (SG) application programming interface category; and a functionality (Fn) application programming interface category.

The resource manager (RM) application programming interfaces help to manage resources on the underlying multimedia data processing system 10 during an MPEG-J session. These resource manager (RM) application programming interfaces can be used to adapt an MPEG session to a given data processing system. The main components of the resource manager (RM) application programming interfaces are the resource manager interface and the event model defined by the resource manager.

The resource manager interface supports a number of different functionalities, including access to the media decoder associated with a particular node, providing a list

of available media decoders of a specific type, access to the priority associated with a media decoder, event models with downloadable event handlers, events mechanisms for stream and decoder problems, and events mechanisms for renderer problems. That is, the resource manager interface provides access to the renderer, the media decoders and their priorities. The resource manager interface also defines methods to gain access to a media decoder given a node in the scene graph and change it to another available media decoder. The media decoders and the decoder functionality application programming interfaces rely on the resource manager to obtain the instance of a media decoder.

The resource manager interface provides a centralized facility for managing resources. The resource manager interface is a collection of a number of classes and interfaces, including MPDecoderEventGenerator, MPRendererEventGenerator, MPDecoderMediaListener, and MPRendererMediaListener interfaces, and "MPDecoderMediaEvents", "ResourceManager", "Renderer", and "MPRendererMediaEvents" classes.

For each media decoder, the resource manager (RM) application programming interface has an installation of a class that implements MPDecoder or a sub-interface. These media decoder installations generate the events for different situations. The resource manager (RM) application programming interface can handle events if necessary in addition to the event handlers in the application. The MPEG-J application engine can receive the Event handlers as bytecode in the bit stream.

Apart from implicitly specifying this event model, the resource manager (RM) application programming interface also provides access to the media decoders and their priorities. Given a node in the scene graph, resource manager (RM) application programming interface provides access to the media decoder associated with that node. The resource manager (RM) application programming interface also facilitates setting and getting the media decoder priorities. The resource manager (RM) application programming interface also enables changing which media decoder is associated with a particular node.

The resource manager (RM) application programming interfaces also define decoder and renderer events. The corresponding interfaces for event generators and

listeners, such as event handlers, are also defined. The resource manager (RM) application programming interfaces implicitly define an event model for graceful degradation. For each media decoder, the resource manager (RM) application programming interfaces would have an instantiation of a class that implements an MPDecoder interface or a sub-interface. These decoder instances generate the events corresponding to different situations. These events can handle events if necessary in addition to the default event handlers in the application. The MPEG-J application engine can receive the event handlers as bytecode in the bitstream.

Program execution may be contingent upon the configuration of the particular data processing system and its capabilities. An MPEG-J session may need to be aware of its environment, so that it is able to adapt its own execution, and the execution of the various components as they may be configured and running in the MPEG-4 browser 100.

The network (Nw) application programming interfaces allow control of the network component of the MPEG-4 browser or presentation engine 100. Through the network (Nw) application programming interfaces, the MPEG-J application engine can interact with the distributed network to which the MPEG-4 browser or presentation engine 100 is connected. The network (Nw) application programming interfaces allow the MPEG-J application engine to monitor the network resources and controllably degrade the multimedia scene in view of the network resources. It should be appreciated that those skilled in the art would be able to provide network (Nw) application programming interfaces for the MPEG-4 browser or presentation engine 100 and the MPEG-J application engine described herein using known techniques and programming structures.

The media decoder (MD) application programming interfaces facilitate basic control, such as start, stop, pause, resume and the like, of all the installed decoders in an MPEG session. A list of decoders that are currently installed and those available can be obtained through the resource manager. The resource manager also provides the instance of the decoder associated with a node in the scene graph.

The media decoder (MD) application programming interfaces support a number of different functionalities, including starting, stopping, pausing, and resuming a decoder,

attaching and detaching a decoder to elementary streams, and getting the type and other attributes of a decoder.

The MPDecoder is an interface that abstracts the most generic decoder. The MPDecoder interface facilitates attaching and detaching decoders to elementary streams. It also provides for access to the type, vendor, and an instance number of the decoder. A number of interfaces that abstract decoders, like a scalable video decoder, i.e., the SclVideoDecoder interface, a structured audio decoder, i.e., the SclAudioDecoder interface, that are sub-interfaces of MPDecoder interface are also defined. These subinterfaces provide access and control to specialized features of that particular decoder.

The scene graph (SG) application programming interfaces provide mechanisms for controlling the scene graph and/or the scene graph controller. The scene graph (SG) application programming interface allows the behavior of a node of the scene graph to be changed and allows the scene graph to be edited. The scene graph (SG) application programming interface supports a Browser Object. The scene graph (SG) application programming interface also supports updatable nodes and/or MPEG-J accessible nodes of the scene graph.

For the content provider scenario described above, the scene graph (SG) application programming interface supports user priority only on those objects that the content provider has indicated as allowable, and scene graph manipulation only corresponding to those objects that the content provider has indicated as allowable. For the user scenario described above, the scene graph (SG) application programming interface supports user priority and scene graph manipulation on all objects. On exemplary embodiment of the scene graph (SG) application programming interface allows for tagging multiple streams as sources and selecting one stream as the master.

In the scene graph (SG) application programming interface according to this invention, there is a browser for each subtree, and is the interface between MPEG-J and that subtree. There are two modes for updating a node, a binary format of scene stream mode and a direct node access mode. The scene graph (SG) application programming interface according to this invention supports binary format for scene updates, turning off the rendering of certain objects, attaching objects to nodes, and the like.

The functionality (Fn) application programming interfaces provide mechanisms for controlling the media decoding process. In particular, a number of useful predefined functionalities become possible under user control. The functionality (Fn) application programming interfaces are dependent on other parts of the other parts of the MPEG-J application engine 200, such as the media decoder (MD) application programming interfaces, the resource manager (RM) application programming interfaces, and the scene graph (SG or SG-EAI) application programming interfaces. Accordingly, the functionality (Fn) application programming interfaces will most likely change as these other application programming interfaces mature.

The current implementation of the functionality (Fn) application programming interfaces support a number of different functionalities, including a ProgressiveImage function; a HotObject function; a DirecDecoding function; a Transparency function; and a TrickDecoding function.

The functionality (Fn) application programming interfaces include interfaces for each of these functionalities. The ProgressiveImage interface triggers the action of progressive refinement of quality of the image being decoded under user control. The ProgressiveImage interface extends the "SclTextureDecoder", which is used to decode image texture streams. The HotObject interface triggers the action of enhancement of a visual object provided that the object is a hot object. Thus hot objects have some enhancement streams associated with them that are triggered when needed. The HotObject interface extends the "SclVideoDecoder", which is used to decode base and enhancement streams.

The DirecDecoding interface allows creation of visual objects that are directionally sensitive. The DirecDecoding interface is most easily explained by assuming a bitstream composed of a number of static video object planes (visual Ops) coded as an audio-video object such that depending on the user interaction, video object planes corresponding to one or more viewpoint are decoded as needed. The Transparency interface allows formation of visual objects with transparency information. The Transparency interface also supports selective decoding of objects under user control. The user may not need to decode all objects or portions of object due to

bandwidth/computing resources. If certain objects or portions of object are hidden they may not need to be decoded. The portions of objects not to be shown can be marked with the transparency color, which can be extracted to form transparency information. This process can also be used to highlight or mark objects. The TrickDecoding interface
5 allows selective decoding of audiovisual objects for playback in trick mode (i.e., fast forward (FF) or rewind (FR)) under user control.

Although the MPEG-J systems, methods and application programming interfaces of this invention do not cover a capability manager application programming interface, a profile application programming interface, a terminal capabilities application
10 programming interface, a section filter application programming interface, a service information/object content information application programming interface, or a timing application programming interface, these application programming interfaces are fully described in the MPEG-J standard, and are specifically described in "Text for VM 6.0 Systems Version 2 (MPEG-J)", ISO/IEC JTC1/SC29/WG11/W2741p3, Seoul, Korea,
15 March 1999. The network application programming interface is described in greater detail in "Text for ISO/IEC 14496-1/PDAMI (MPEG-4 version 2 MPEG-J)", ISO/IEC JTC1/SC29/WG11/N2739p3, Seoul, Korea, March 1999.

The MPEG-J application programming interface specifications for the various MPEG-J application programming interface described above are set forth below.

20 The MPEG-J Application Manager application programming interface specification is set forth below:

Interface mpgj.sys.ApplicationManager

public interface **ApplicationManager**

This interface allows the scene graph to access information from application.

25 This interface is implemented by the application, the application manager is registered with a node that can call the application.

Constructors

public **ApplicationManager()**

Methods

```
Public java.lang.String getProperty(int nodeId,
java.lang.String property)
```

5 This method will be called by node object to get property information from the application or applet. The node identifies itself with nodeId, while the string specifies the property that it is interested in.

Class mpgj.sys.NodeCanCallAM

```
java.lang.Object
|
10 +----vrml.external.Node
|
+----mpgj.sys.NodeCanCallAM
```

```
public abstract class NodeCanCallAM
extends vrml.external.Node
```

15 This is a special node object that is capable of calling Application Manager.

Constructors

```
public NodeCanCallAM()
```

Methods

```
public void registerAM(ApplicationManager am)
```

20 This is to be called by MPEG-J application.

The MPEG-J Scene Graph application programming interface specification is set forth below:

Interface mpgj.sys.ScenegraphManager

```
public interface ScenegraphManager
```

25 Interface that MPEG-J applications access and manipulate the BIFS player.

Methods

```
public void createNode(java.lang.String nodeName,
                        java.lang.String[] fields,
                        java.lang.String[] values)
```

30 Creates a node with nodeName, fields, and field values. There is a one to one correspondence between the field names and the values and the field names and the values are given in the same order. If a field is missing, the default value is used as

specified in the systems document. The underlying implementation is expected to create a node exactly the same way in which it is done in the case of creating a node in the BIFS stream.

Parameters:

5 nodeName - The name of the node
 fields - field names
 values - values of fields

```
10            public void addRoute(vrml.external.Node fromNode,  
                                 java.lang.String eventOut,  
                                 vrml.external.Node toNode,  
                                 java.lang.String eventIn).
```

Adds a route between two nodes, from an eventOut to an eventIn. If the ROUTE already exists, this method silently exits. It does not attempt to add a second parallel ROUTE.

```
15            public void deleteRoute(vrml.external.Node fromNode,  
                                     java.lang.String eventOut,  
                                     vrml.external.Node toNode,  
                                     java.lang.String eventIn)
```

Deletes a route between two nodes. If the route does not exist, the method silently exits.

```
20            public void beginUpdate()
```

Locks the output from the external interface to the scene graph as the code is about to begin a series of updates. No events will be passed to the scene graph. They will be buffered pending release due to a subsequent call to endUpdate. This call is a nesting call which means subsequent calls to beginUpdate are kept on a stack. No events will be released to the scene graph until as many endUpdates have been called as beginUpdate.

```
25            public void endUpdate()
```

Releases the output of events from the external interface into the scene graph. All events posted to this point from the last time that beginUpdate was called are released into the scene graph for processing at the next available opportunity. This call is a

nesting call which means subsequent calls to `beginUpdate` are kept on a stack. No events will be released to the scene graph until as many `endUpdates` have been called as `beginUpdate`. If no `beginUpdate` has been called before calling this method, it has no effect.

```
5      public vrml.external.Node getNode(java.lang.String
                                         name)
```

Gets a DEF node by name. Nodes given DEF names in the root scene graph are available to be retrieved by this method. DEFed nodes in Inlines are not available.

10 Although they have not been explicitly included here, all the EAI API's except the Browser interfaces and classes of the Virtual Reality Markup Language can be used for scene graph control.

The MPEG-J Resource Manager application programming interface specification is set forth below:

Interface mpgj.sys.MPDecoderEventGenerator

15 Subinterfaces: MPDecoder, SclDecoder

public interface MPDecoderEventGenerator

Methods

```
public void addMPDecoderMediaListener
(MPDecoderMediaListener l)
```

```
20     public void removeMPDecoderMediaListener
        (MPDecoderMediaListener l)
```

Interface mpgj.sys.MPRendererEventManager

Subinterfaces: Renderer

public interface MPRendererEventManager25 **Methods**

```
public void addMPRenderMediaListener
(MPRenderMediaListener l)
```

```
public void removeMPRendererMediaListener
(MPRenderMediaListener l)
```

Interface mpgj.svs.Renderer**public interface Renderer****implements MPRendererEventGenerator**

For each decoder, the Resource Manager would have an installation of a class that implements the MPDecoder interface, described below, or a sub-interface. These decoder installations generate the events for different situations. The resource manager implementation can handle events, if necessary, in addition to the event handlers in the application (the order of which is left to the implementation). The MPEG-J application can receive the Event handlers as bytecode in the bitstream.

Interface mpgj.svs.ResourceManager**public interface ResourceManager****Methods****public void setDecPriority(MPDecoder dec, int priority)**

Overrides or changes the priority of a media elementary stream.

public int getDecPriority(MPDecoder dec)

Gets the priority of a media elementary stream.

**public void checkDynResources(int memory, int
scratch, int processLoad)**

Gets dynamic system resource information, such as, for example, information about memory, scratch space and processor loads.

**public void checkStatResources(int grpCards, int
devices, int plugins)**

Gets static system resource information, such as, for example, information about graphics cards, hardware devices and software plugins.

public MPDecoder getDecoder(int nodeId)

Gets the media decoder for the identified node.

public Renderer getRenderer()

Gets the renderer and readies the presentation engine to add events to renderer.

Interface mpgj.sys.MPDecoderMediaListener

public interface MPDecoderMediaListener

extends java.util.EventListener

Methods

5 public abstract void mpDecoderMediaHandler
(MPDecoderMediaEvents event)

Interface mpgj.sys.MPRendererMediaListener

public interface MPRendererMediaListener

extends java.util.EventListener

10 **Methods**

public abstract void mpRendererMediaHandler
(MPRendererMediaEvents event)

Class mpgj.sys.MPDecoderMediaEvents

15 java.lang.Object ,
 |
 +----java.util.EventObject
 |
 +----mpgj.sys.MPDecoderMediaEvents

public class MPDecoderMediaEvents

20 extends java.util.EventObject

Constants and Variables

public static final int DECODER_OVERFLOW
public static final int DECODER_SYNC_ERROR
public static final int STREAM_UNDERFLOW
25 public static final int STREAM_OVERFLOW
public static final int STREAM_START
public static final int STREAM_END

Constructors

public MPDecoderMediaEvents(int condition)

Methods

```
public int getCondition()
```

The obtained condition can correspond to decoder overflow, sync error or end of decode.

5 **Class mpgj.sys.MPRendererMediaEvents**

```

        java.lang.Object
            |
            +----java.util.EventObject
                    |
10         +----mpgj.sys.MPRendererMediaEvents

```

```
public class MPRendererMediaEvents
```

```
    extends java.util.EventObject
```

Constants and Variables

```

15     public static final int DECODER_UNDERFLOW
        public static final int MISSED_FRAMES

```

Constructors

```
public MPRendererMediaEvents(int condition)
```

Methods

```
20     public int getCondition()
```

The MPEG-J Media Decoders application programming interface specification is set forth below:

Interface mpgj.dec.SclDecoder

Subinterfaces: SclTextureDecoder, SclVideoDecoder

```
25     public interface SclDecoder
```

```
        extends MPDecoderEventGenerator, MPStreamEventGenerator
```

This is the base interface for scalable enhancement.

Methods

```

        public int getLevel()
        Gets the spatial enhancement layer number.
30     public int getMode()
        Gets the spatial enhancement mode number.

```

```
public boolean sptEnh(int e_level, e_mode)
    throws InvalidDecoderLevelException,
InvalidDecoderModeException
```

5 Sets the decoding mode to spatial enhancement layer and returns a boolean to indicate if it succeeded or not, e_level is the enhancement layer number, while e_mode is the subtype in that layer.

```
public boolean tmpEnh(int e_level, int e_mode)
    throws InvalidDecoderLevelException,
InvalidDecoderModeException
```

10 Sets the decoding mode to temporal enhancement layer and returns a boolean to indicate if it succeeded or not.

```
public boolean snrEnh(int e_level, int e_mode)
    throws InvalidDecoderLevelException,
InvalidDecoderModeException
```

15 Sets the decoding mode to snr enhancement layer and returns a boolean to indicate if it succeeded or not.

Interface mpgj.dec.VideoDecoder

Subinterfaces: SclVideoDecoder, Transparency

```
public interface VideoDecoder
20       extends MPDecoder
```

Decoding of video objects.

Interface mpgj.dec.SclVideoDecoder

Subinterfaces: DirecDecoding, HotObject

```
public interface SclVideoDecoder
25       extends VideoDecoder, SclDecoder
```

Decoding of scalable video objects.

Methods

```
public boolean dataPart(int e_level, int e_mode)
30       Sets the decoding mode to data partitioning enhancement layer and returns a boolean to indicate if it succeeded or not.
```

Interface mpgj.dec.TextureDecoder

Subinterfaces: SclTextureDecoder

public interface **TextureDecoder**

extends MPDecoder

5 Decoding of image texture objects

Interface mpgj.dec.SclTextureDecoder

Subinterfaces: ProgressiveImage

public interface **SclTextureDecoder**

extends TextureDecoder, SclDecoder

10 **Interface mpgj.dec.MeshDecoder**public interface **MeshDecoder**

extends MPDecoder

Decoding of wireframe mesh objects

Interface mpgj.dec.FaceDecoder

15 Subinterfaces: SclFaceDecoder

public interface **FaceDecoder**

extends MPDecoder

Decoding of Face objects

Interface mpgj.dec.SclFaceDecoder20 public interface **SclFaceDecoder**

extends FaceDecoder

This interface extends FaceDecoder. It contains methods to scale decoder performance.

Methods

25 public boolean pickModel(int mode)

Sets the decoding mode and returns a boolean to indicate if it succeeded or not.

The 'mode' variable indicates one of the three choices - default model without calibration, default model with calibration, and, the downloaded model.

```
public boolean pickFAP(int mode)
```

Sets the decoding mode and returns a boolean to indicate if it succeeded or not.

The 'mode' variable indicates one of the two choices - all FAP groups, and, viseme and expression group of FAP only.

```
5      public boolean pickFIT(int mode)
```

Sets the decoding mode and returns a boolean to indicate if it succeeded or not.

The 'mode' variable indicates one of the two choices - use FIT, and, ignore FIT.

```
public boolean sclBitrateDown()
```

Decrease decoder bitrate.

```
10     public boolean sclComplexity()
```

Decrease decoder complexity.

Interface mpgj.dec.AudioDecoder

```
public interface AudioDecoder
```

```
    extends MPDecoder
```

```
15     Interface mpgj.dec.SclAudioDecoder
```

```
public interface SclAudioDecoder
```

```
    extends AudioDecoder
```

This interface extends AudioDecoder. It contains methods to scale decoder performance.

```
20     Constants
```

```
public static final int PARAMETRIC
```

```
public static final int CELP
```

```
public static final int AAC
```

Methods

```
25     public boolean sclBitrateUp ()
```

Increase decoder bitrate for scalable stream.

```
public boolean sclBitrateDown ()
```

Decrease decoder bitrate for scalable stream.

```
public boolean sclBandwidth (int region)
```

```
30     public boolean sclComplexity ()
```



```
public boolean sclType (int type)
```

Interface mpgj.dec.TTSDecoder

```
public interface TTSDecoder
```

```
    extends MPDecoder
```

5 Decoding of TTS objects.

Interface mpgj.dec.StructAudioDecoder

```
public interface StructAudioDecoder
```

```
    extends MPDecoder
```

Decoding of structured audio objects.

10 The MPEG-J Functionality application programming interface specification is set forth below:

Interface mpgj.func.ProgressiveImage

```
public interface ProgressiveImage
```

```
    extends SclTextureDecoder
```

15 ProgressiveImage allows progressive refinement of quality of an image under user control. The images are assumed to be static (still image vops).

Methods

```
public int selectProgLevel ()
```

20 Select level up to which decoding of transform (DCT or wavelet) coefficients will take place. A level constitutes coefficients up to a certain position in scan order.

Interface mpgj.func.HotObject

```
public interface HotObject
```

```
    extends SclVideoDecoder
```

25 HotObject is a interface that triggers the action of enhancement of a visual object provided that the object is a hot object.

Methods

```
public int selectHotType ()
```

Selects type of enhancement (spatial, quality, temporal etc.).

Interface mpgj.func.DirecDecoding**public interface DirecDecoding****extends SclVideoDecoder**

5 DirecDecoding is an interface that allows creation of visual objects that are directionally sensitive.

Methods**public int selectDirec ()**

10 Selects the direction (scene orientation). A number of prespecified directions are allowed and selection takes place by clicking a mouse on hot points on the object or via a menu.

Interface mpgj.func.Transparency**public interface Transparency****extends VideoDecoder**

15 The Transparency interface allows formation of visual objects with transparency information.

Methods**public int redirect()****Interface mpgj.func.TrickDecoding****public interface TrickDecoding**

20 **extends MPDecoder**

Selective decoding of objects for playback intrickmode (FF, FR..) under user control. TrickDecoding is an interface that can be used to decode audio visual objects for trick play.

Methods

25 **public float fastforw (float speed)**

throws InvalidDecoderSpeedException

Start decoding of data at the speed specified. The speed must be positive. The speed = 1 for normal playing. It returns the effective speed set.

public float fastrev (float speed)

30 **throws InvalidDecoderSpeedException**

Stop decoding of data at the speed specified. The speed must be positive. The speed = 1 for normal playing.

```
public void freezeDec ()
```

Temporarily suspend decoding of data.

```
5 public void resumeDec ()
```

Restart decoding of data from current state of pause.

```
public void loopDec ()
```

This method allows user interactive decoding of selected portions of the srcstream for forward or reverse playback at a variety of speeds.

```
10 public float getSpeedAndDirec ()
```

Get the direction of decoding and speed.

Interface mpgj.func.AdjustAudio

```
public interface AdjustAudio
```

```
    extends SclAudioDecoder
```

```
15 AdjustAudio allows refinement of quality of audio under user control.
```

Methods

```
public boolean incrSpeed(float percent)
```

Increase speed of decoder without changing pitch.

```
public boolean decrSpeed(float percent)
```

```
20 Decrease speed of decoder without changing pitch.
```

```
public boolean incrPitch(float percent)
```

Increase pitch of decoder without changing speed.

```
public boolean decrPitch(float percent)
```

Decrease pitch of decoder without changing speed.

```
25 The MPEG-J Device application programming interface specification is set forth  
below:
```

Interface mpgj.dev.InputDevice

```
public interface InputDevice
```

```
30 The input device application programming interface is a common interface for  
input device drivers. The application would expect all of the methods of the application
```

programming interface to be implemented. This API is modeled after the JAVA 3D Input Device application programming interface, and is intended as a starting point for discussion.

Constants

5 public static final int POLLED
 public static final int STREAMING

These flags specify the mode in which the associated device works.

Methods

 public abstract boolean initialize()
10 Returns true if initialization succeeds, false otherwise.
 public abstract void setProcessingMode(int mode)
 public abstract int getProcessingMode()
 public abstract int getSensorCount()

Returns the number of sensor objects associated with the device.

15 public abstract Sensor getSensor(int sensorIndex)
 Returns the specified sensor
 public abstract void

setNominalPositionAndOrientation()

Sets the devices current position and orientation as nominal.

20 public abstract void pollAndProcessInput()
 Polls the device for data and processes the received values.
 public abstract void processStreamInput()
 public abstract void close()

Interface mpgj.dev.AudioDevice

25 public class AudioDevice

The audio device interface is a preliminary specification of what would be required to incorporate a device driver for an audio device into an MPEG-J implementation. This API is modeled after the JAVA 3D AudioDevice API.

Constants

30 public final static int HEADPHONES
 public final static int MONO_SPEAKER

```
public final static int STEREO_SPEAKERS
public final static int SURROUND_5_1
public final static int SURROUND_7_1
```

Methods

```
5      public abstract boolean initialize ()
      public abstract boolean close ()
      public abstract void setAudioPlaybackType (int type)
      public abstract int getAudioPlaybackType ()
      Set or retrieve the device type (Mono Stereo or Headphones)
10     public abstract void setCenterEarToSpeaker (float
distance)
      public abstract float getCenterEarToSpeaker ()
      Sets or retrieves the distance from the center ear and one of the speakers in the
listeners environment.
15     public abstract void setAngleOffsetToSpeaker (float
angle)
      public abstract float getAngleOffsetToSpeaker ()
      Sets or retrieves the angle between the vectors from the center ear to each of the
speaker transducers.
20     public abstract int getTotalChannels ()
      public abstract int getChannelsAvailable ()
      public abstract int getChannelsUsedForSound ()
      Returns the number of channels needed to render a particular sound node.
```

25 The MPEG-J Interactivity Manager application programming interface
specification is set forth below:

Interface mpgj.sys.InteractivityManager

```
public interface InteractivityManager
```

This interface provides a centralized facility for managing user interactivity with
the scene.

Methods

```
public java.lang.String getName()  
public void  setPickability(int id, int pickability)  
public void  clearPickability(int id, int pickability)  
5 public void  getPickability(int id, int pickability)  
public void  resetPickability(int id)
```

The value "id" specifies the node for every pickable object. There is 1 bit associated with each capability. Thus, it is possible to set or reset one capability at a time. The Mouse Events include mouse click, mouse drag, mouse entered, mouse exited,
10 mouse first, mouse last, mouse moved, mouse pressed, and mouse released.

As shown in Figs 1-5, the MPEG-J application engine and the MPEG-4 browser are preferably implemented on a multimedia data processing system such as a programmed general purpose computer. However, the MPEG-J application engine and the MPEG-4 browser can also be implemented on a special purpose computer, a programmed
15 microprocessor or microcontroller and peripheral integrated circuit elements, an ASIC or other integrated circuit, a digital signal processor, a hardwired electronic or logic circuit such as a discrete element circuit, a programmable logic device such as a PLD, PLA, FPGA or PAL, or the like. In general, any device, capable of implementing a finite state machine that is in turn capable of implementing the application programming interfaces and the
20 various system elements shown in Figs. 1-5, can be used to implement the MPEG-J application engine and the MPEG-4 browser.

While this invention has been described in conjunction with the exemplary embodiments outlined above, it is evident that many alternatives, modifications and variations will be apparent to those skilled in the art. Accordingly, the exemplary
25 embodiments of the invention, as set forth above, are intended to be illustrative, not limiting. Various changes may be made without departing from the spirit and scope of the invention.

WHAT IS CLAIMED IS:

- 1 1. A multimedia browsing system, comprising:
2 a multimedia browsing subsystem that generates a multimedia scene based
3 on a scene description, the multimedia scene comprising at least one multimedia object
4 and the scene description defining spatial and temporal positioning of each object within
5 the scene; and
6 a multimedia browsing control subsystem that controls the multimedia
7 browsing subsystem based on a control program associated with the at least one
8 multimedia object.

- 1 2. The multimedia browsing system of claim 1, wherein the multimedia
2 browsing control subsystem controls a behavior of the multimedia browsing subsystem
3 based on behavior control commands of the control program associated with the at least
4 one multimedia object.

- 1 3. The multimedia browsing system of claim 2, wherein:
2 the multimedia browsing system is executing on a data processing
3 apparatus having at least one system resource; and
4 the behavior control commands associated with the at least one
5 multimedia object controllably degrade the multimedia scene based on a status of at least
6 one of the at least one system resource.

- 1 4. The multimedia browsing system of claim 3, wherein the at least one
2 system resource is a capacity of a communications channel over which the data
3 processing apparatus receives at least one data stream corresponding to the at least one
4 multimedia object.

- 1 5. The multimedia browsing system of claim 4, wherein the communications
2 channel is at least one of a wireless channel and a channel of a distributed network.

1 6. The multimedia browsing system of claim 4, wherein the capacity of the
2 communications channel is at least one of limited and changing over time.

1 7. The multimedia browsing system of claim 3, wherein each system
2 resource is one of a memory resource, a processing resource, a display resource, and a
3 network resource.

1 8. The multimedia browsing system of claim 3, wherein the multimedia
2 browsing subsystem is controlled by the multimedia browsing control subsystem through
3 at least one application programming interface.

1 9. The multimedia browsing system of claim 8, wherein at least one
2 application programming interface of the at least one application programming interface
3 is an object-oriented program.

1 10. The multimedia browsing system of claim 9, wherein eat least one
2 application programming interface of the at least one application programming interface
3 is a Java program.

1 11. The multimedia browsing system of claim 8, wherein:
2 the multimedia browsing subsystem comprises at least one buffer, each
3 buffer storing data corresponding to at least one of the at least one object; and
4 at least one application programming interface of the at least one
5 application programming interface is used by the multimedia browsing control subsystem
6 to control the at least one buffer of the multimedia browsing subsystem based on the
7 behavior control commands.

1 12. The multimedia browsing system of claim 8, wherein:
2 the multimedia browsing subsystem comprises a compositor, the
3 compositor combining data corresponding to at least one of the at least one object based
4 on the scene description to generate the multimedia scene; and

5 at least one application programming interface of the at least one
6 application programming interface is used by the multimedia browsing control subsystem
7 to control the compositor of the multimedia browsing subsystem based on the behavior
8 control commands.

1 13. The multimedia browsing system of claim 1, wherein the scene description
2 defines at least one dynamic property of at least one object.

1 14. The multimedia browsing system of claim 1, wherein the multimedia
2 browsing subsystem is controlled by the multimedia browsing control subsystem through
3 at least one application programming interface.

1 15. The multimedia browsing system of claim 14, wherein each at least one
2 application programming interface is an object-oriented program.

1 16. The multimedia browsing system of claim 15, wherein each at least one
2 application programming interface is a Java program.

1 17. The multimedia browsing system of claim 14, wherein the at least one
2 application programming interface includes at least one of an application manager
3 application programming interface, a scene graph application programming interface, a
4 resource manager application programming interface, a media decoders application
5 programming interface, a functionality application programming interface, a network
6 application programming interface, a device application programming interface, and an
7 interactivity application programming interface.

1 18. The multimedia browsing system of claim 17, wherein:
2 the multimedia browsing control subsystem comprises:
3 an application controller, and
4 a scene graph manager;
5 the scene graph application programming interface connects the
6 application controller and the scene graph manager; and

7 the scene graph manager is connected to a scene graph controller of the
8 multimedia browsing subsystem.

1 19. The multimedia browsing system of claim 18, wherein the scene graph
2 manager controls the scene graph controller of the multimedia browsing subsystem to
3 modify the scene description.

1 20. The multimedia browsing system of claim 19, wherein:
2 the multimedia browsing system is executing on a data processing
3 apparatus having at least one system resource; and
4 the scene graph manager controls the scene graph controller of the
5 multimedia browsing subsystem to controllably degrade the multimedia scene based on a
6 status of at least one of the at least one system resource.

1 21. The multimedia browsing system of claim 20, wherein the scene manager
2 controls the scene graph controller of the multimedia browsing subsystem to controllably
3 degrade the multimedia scene based on the behavior control commands and the status of
4 at least one of the at least one system resource.

1 22. The multimedia browsing system of claim 19, wherein the scene manager
2 modifies the scene description to controllably degrade the multimedia scene based on at
3 least one of the behavior control commands.

1 23. The multimedia browsing system of claim 19 wherein the scene manager
2 modifies the scene description based on at least one user input.

1 24. The multimedia browsing system of claim 18, wherein
2 the multimedia browsing subsystem further comprises a scene graph
3 decoder that decodes encoded scene graph data, and that outputs the decoded scene graph
4 data to the scene graph manager;

5 the multimedia browsing control subsystem further comprises a scene
6 encoder connected to the scene graph decoder of the multimedia browsing subsystem and
7 controllably connectable to the scene graph manager;
8 the scene graph manager controls the scene encoder to create additional
9 encoded scene graph data, the scene encoder outputting the additional encoded scene
10 graph data to the scene graph decoder.

1 25. The multimedia browsing system of claim 24, wherein the multimedia
2 browsing control subsystem further comprises a switch that selectively connects the scene
3 graph manager to one of the scene encoder and the scene graph manager.

1 26. The multimedia browsing system of claim 17, wherein:
2 the multimedia browsing subsystem comprises:
3 a scene graph controller;
4 at least one decoding buffer,
5 at least one media decoder,
6 at least one composition buffer, and
7 a compositor;
8 the multimedia browsing control subsystem comprises:
9 an application controller, and
10 a resource manager;
11 at least one of the resource manager, media decoder and functionality
12 application programming interfaces connect the application controller and the resource
13 manager; and
14 the resource manager is connected to the scene graph controller, each
15 decoding buffer, each media decoder, each composition buffer and the compositor of the
16 multimedia browsing subsystem.

1 27. The multimedia browsing system of claim 26, wherein the resource
2 manager controls at least one of the scene graph controller, at least one decoding buffer,
3 at least one media decoder, at least one composition buffer and the compositor.

1 28. The multimedia browsing system of claim 27, wherein:
2 the multimedia browsing system is executing on a data processing
3 apparatus having at least one system resource; and
4 the resource manager controls at least one of the scene graph controller, at
5 least one decoding buffer, at least one media decoder, at least one composition buffer and
6 the compositor based on a status of at least one of the at least one system resource.

1 29. The multimedia browsing system of claim 28, wherein the resource
2 manager controls at least one of the scene graph controller, at least one decoding buffer,
3 at least one media decoder, at least one composition buffer and the compositor to
4 controllably degrade the multimedia scene based on the status of at least one of the at
5 least one system resource.

1 30. The multimedia browsing system of claim 29, wherein the resource
2 manager controls at least one of the scene graph controller, at least one decoding buffer,
3 at least one media decoder, at least one composition buffer and the compositor to
4 controllably degrade the multimedia scene based on the behavior control commands
5 associated with the at least one multimedia object and the status of at least one of the at
6 least one system resource.

1 31. The multimedia browsing system of claim 27, wherein the resource
2 manager controls at least one of the scene graph controller, at least one decoding buffer,
3 at least one media decoder, at least one composition buffer or the compositor to
4 controllably degrade the multimedia scene based on the behavior control commands.

1 32. The multimedia browsing system of claim 27, wherein the resource
2 manager controls at least one of the scene graph controller, at least one decoding buffer,
3 at least one media decoder, at least one composition buffer or the compositor based on at
4 least one user input.

- 1 33. The multimedia browsing system of claim 26, wherein:
2 the multimedia browsing control subsystem further comprises an
3 application manager;
4 the resource manager and functionality application programming
5 interfaces connect the application controller and the resource manager; and
6 the media decoder application programming interface connects the
7 application manager and the resource manager.
- 1 34. The multimedia browsing system of claim 17, wherein:
2 the multimedia browsing control subsystem comprises an application
3 controller; and
4 the device application programming interface connects the application
5 controller and at least one input/output device.
- 1 35. The multimedia browsing system of claim 17, wherein:
2 the multimedia browsing control subsystem comprises:
3 an application controller, and
4 an interactivity manager;
5 the interactivity application programming interface connects the
6 application controller and the interactivity manager;
7 the device application programming interface connects the interactivity
8 manager and at least one input/output device.
- 1 36. The multimedia browsing system of claim 1, wherein at least one of the at
2 least one object is one of an audio object, a video object, a synthetic object, a synthetic
3 face object, and a synthetic body object.
- 1 37. A method for browsing an MPEG-4 multimedia scene comprising at least
2 one MPEG-4 multimedia object, comprising:

3 generating the MPEG-4 multimedia scene based on an MPEG-4 scene
4 description, the MPEG-4 scene description defining spatial and temporal positioning of
5 each MPEG-4 multimedia object within the scene; and

6 controlling the generation of the MPEG-4 multimedia scene based on an
7 MPEG-J control program associated with the at least one MPEG-4 multimedia object.

1 38. The method of claim 37, wherein controlling the generation of the
2 MPEG-4 multimedia scene comprises executing at least one behavior control command
3 of the MPEG-J control program associated with the at least one MPEG-4 multimedia
4 object.

1 39. The method of claim 37, wherein:
2 generating the MPEG-4 multimedia scene comprises executing the
3 MPEG-4 scene description on a data processing apparatus having at least one system
4 resource; and
5 controlling the generation of the MPEG-4 multimedia scene comprises
6 controllably degrading the MPEG-4 multimedia scene based on a status of at least one of
7 the at least one system resource.

1 40. A computer program product, comprising a computer usable storage
2 medium containing:

3 first computer readable program code that causes a data processing
4 apparatus to serve as an MPEG-4 browser or presentation engine that generates a
5 MPEG-4 multimedia scene based on an MPEG-4 scene description, the MPEG-4
6 multimedia scene comprising at least one MPEG-4 multimedia object and the MPEG-4
7 scene description defining spatial and temporal positioning of each MPEG-4 multimedia
8 object within the MPEG-4 multimedia scene; and

9 second computer readable program code that causes the data processing
10 apparatus to serve as an MPEG-J application engine that controls the MPEG-4 browser or
11 presentation engine based on an MPEG-J control program associated with the at least one
12 MPEG-4 multimedia object.

1 41. The computer program product of claim 40, wherein the MPEG-J
2 application engine controls a behavior of the MPEG-4 browser or presentation engine
3 based on behavior control commands of the MPEG-J control program associated with the
4 at least one MPEG-4 multimedia object.

1 42. The computer program product of claim 41, wherein:
2 the data processing apparatus has at least one system resource; and
3 the behavior control commands associated with the at least one MPEG-4
4 multimedia object controllably degrade the MPEG-4 multimedia scene based on a status
5 of at least one of the at least one system resource.

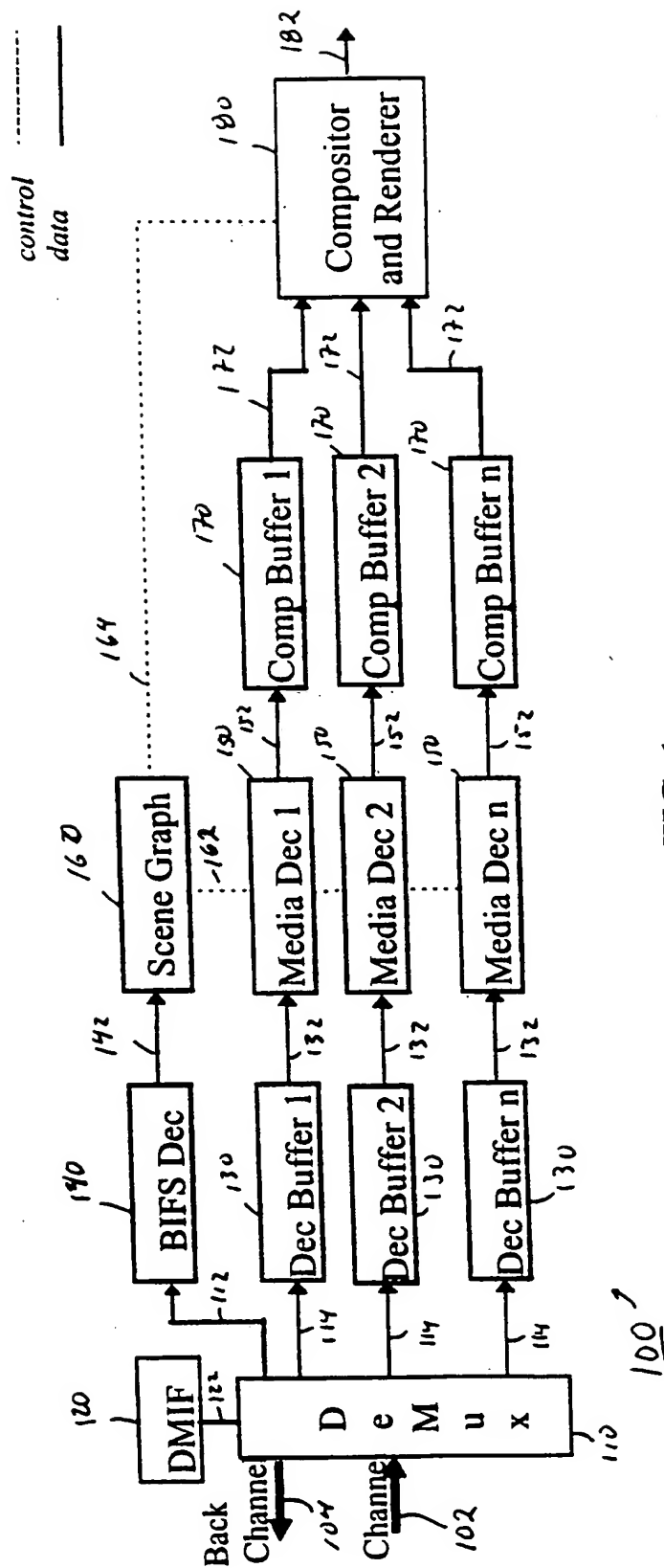


FIG. 1

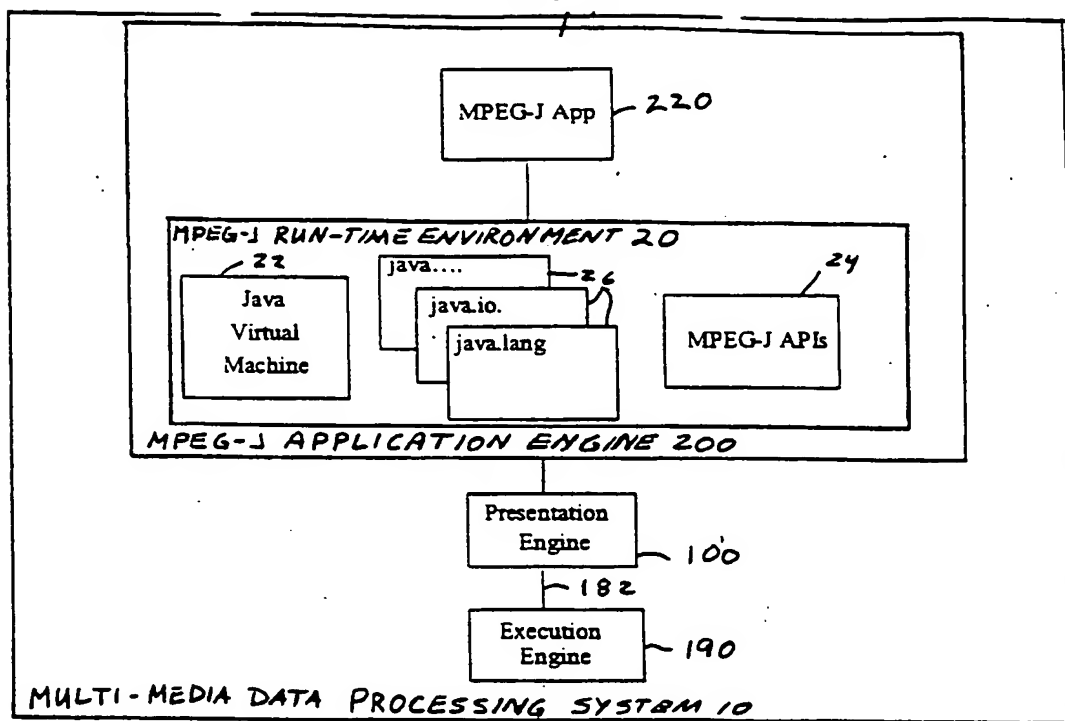


FIG. 2

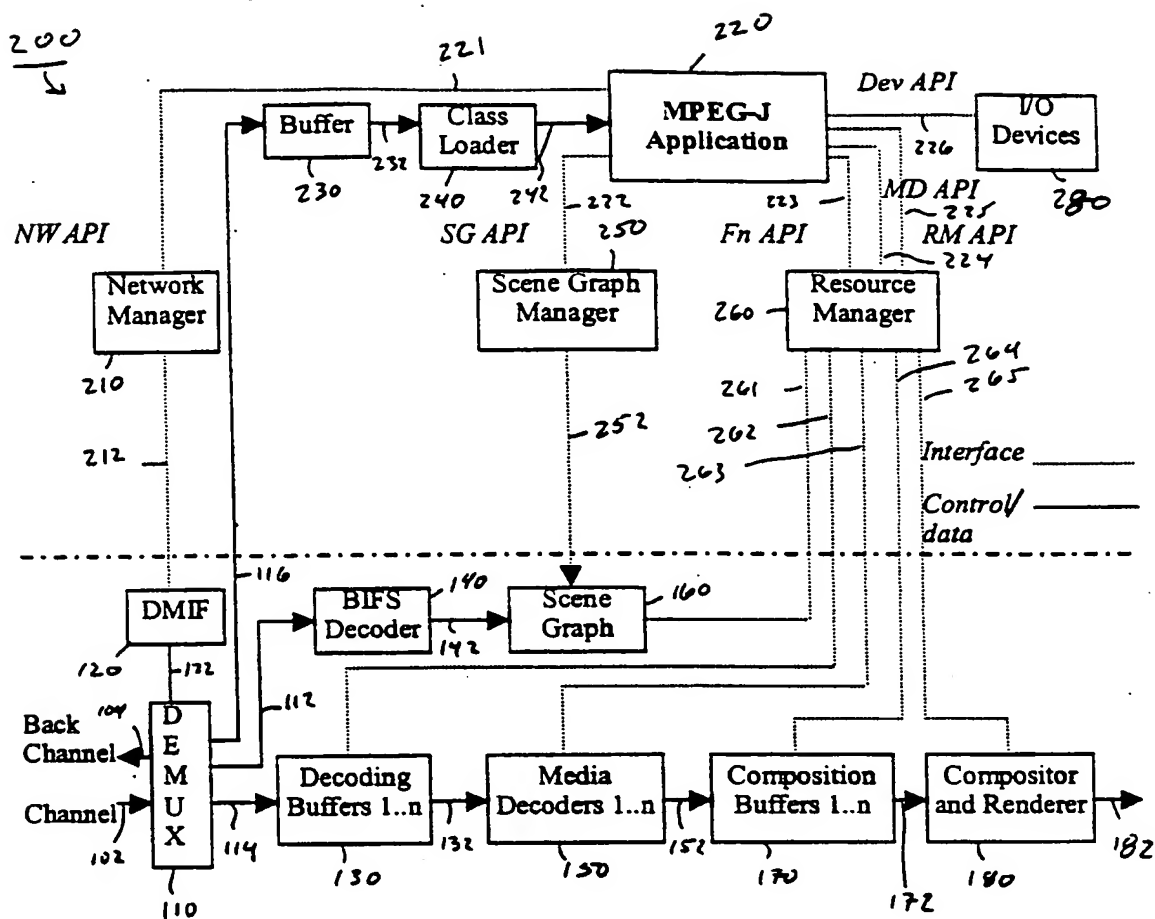


FIG. 3

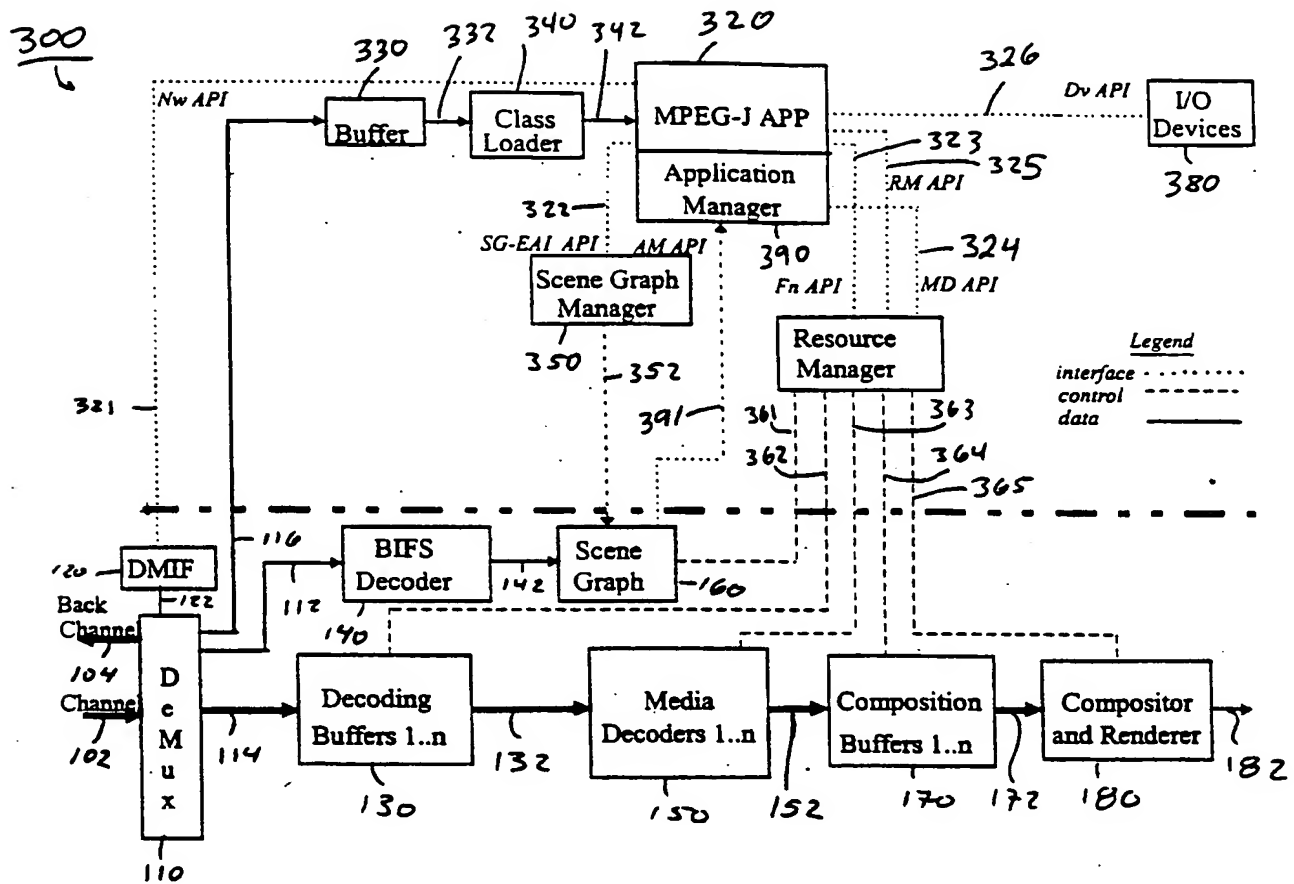


FIG. 4

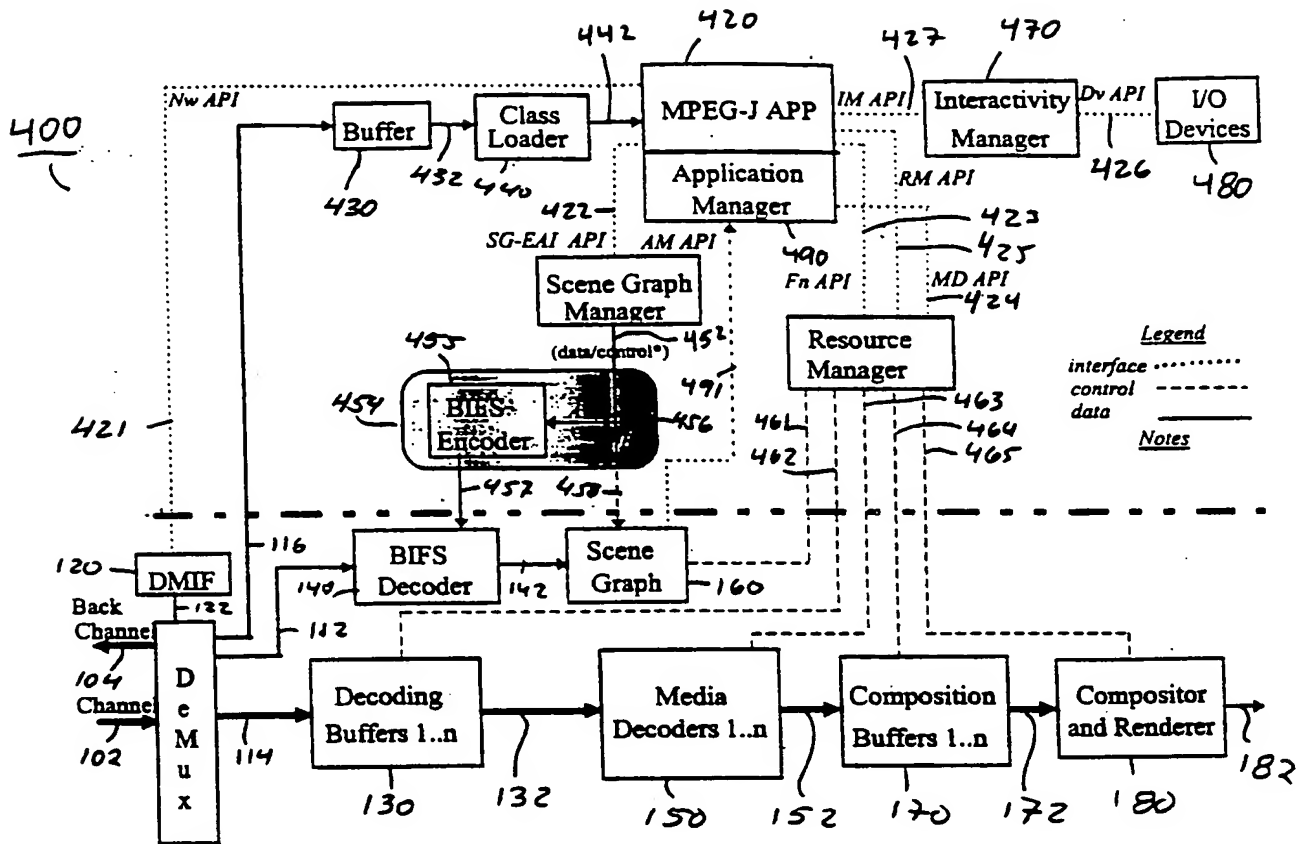


FIG. 5

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record.**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☒ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☒ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.